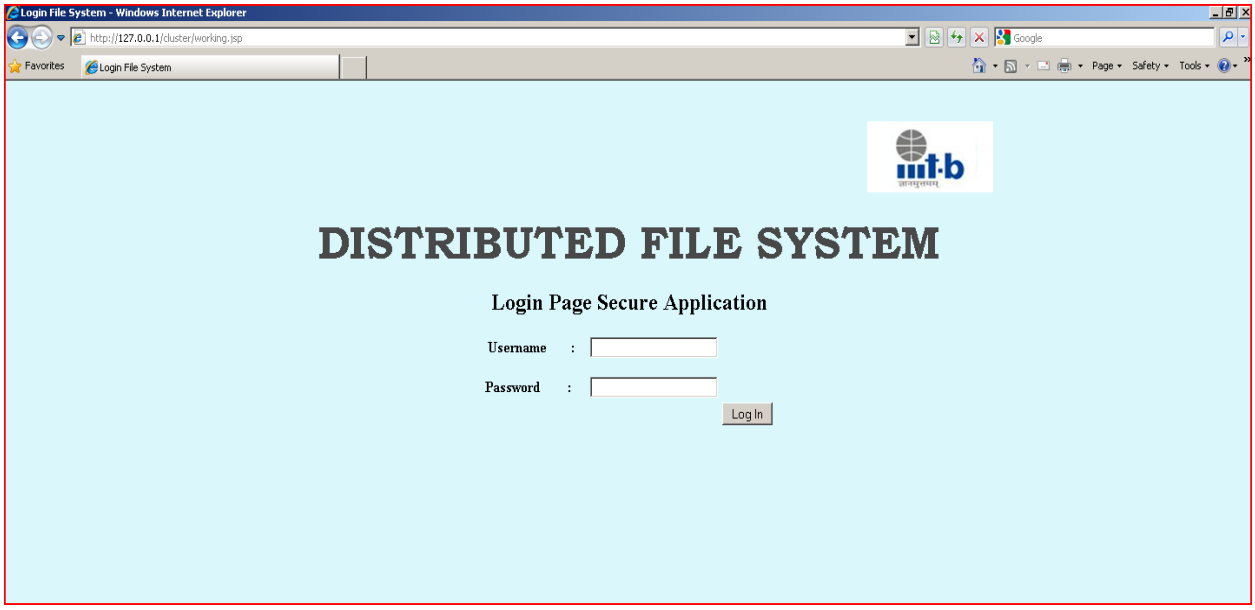
User Manual for the Distributed File System:

The following steps guide a user through the features of DFS:

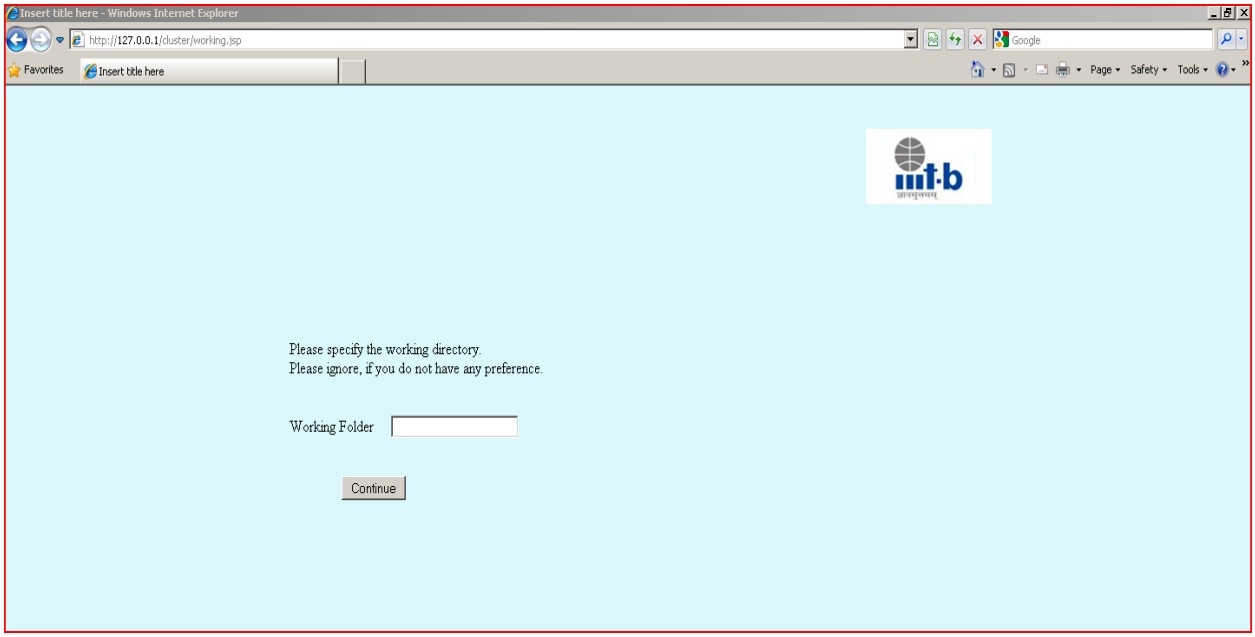
The Initial login screen appears upon start of DFS.

1. The login Screen appears :

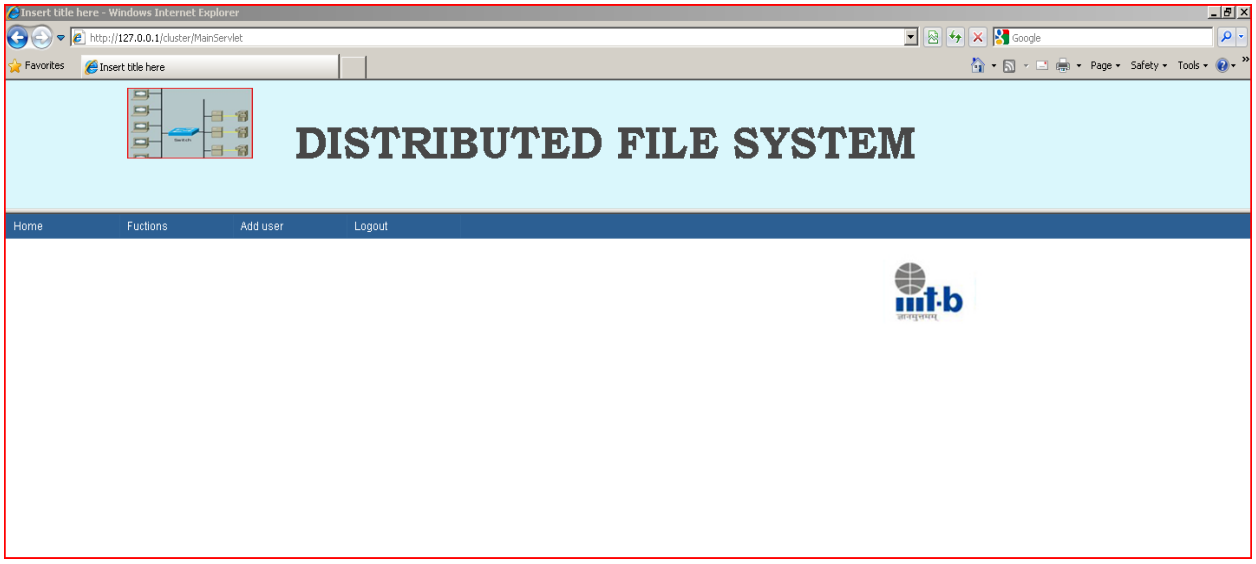
   Enter authorized credentials and click on **Log In** Button.



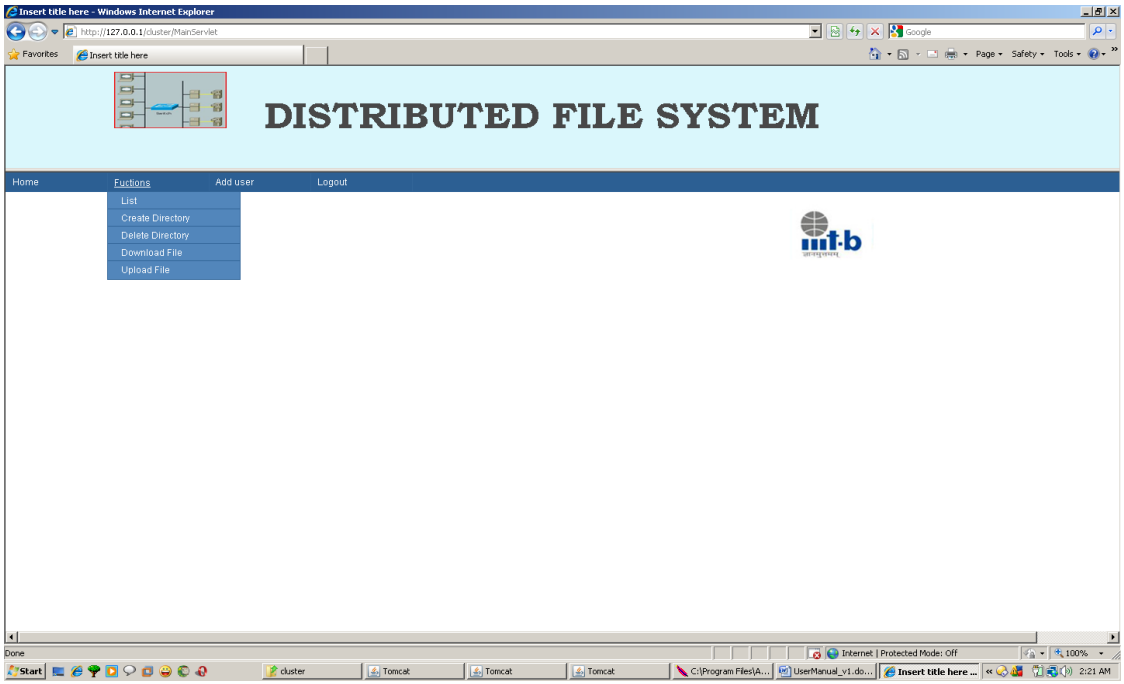
2. The following Home Screen appears after authorization.



3. After clicking Continue this screen appears

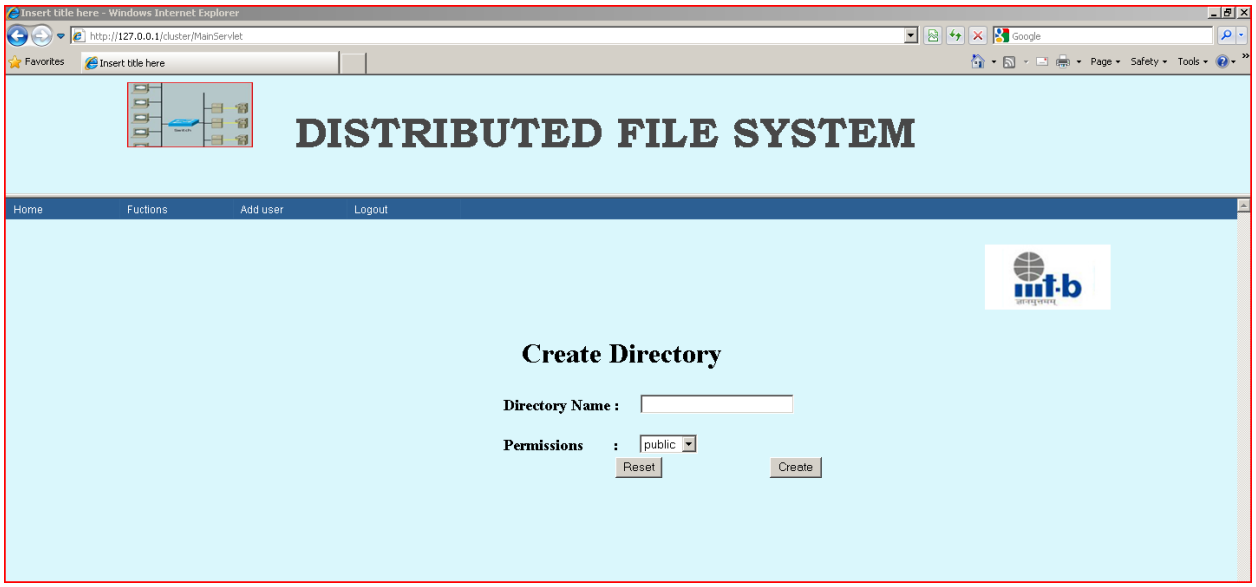


4. After placing pointer on functions the following options appear :



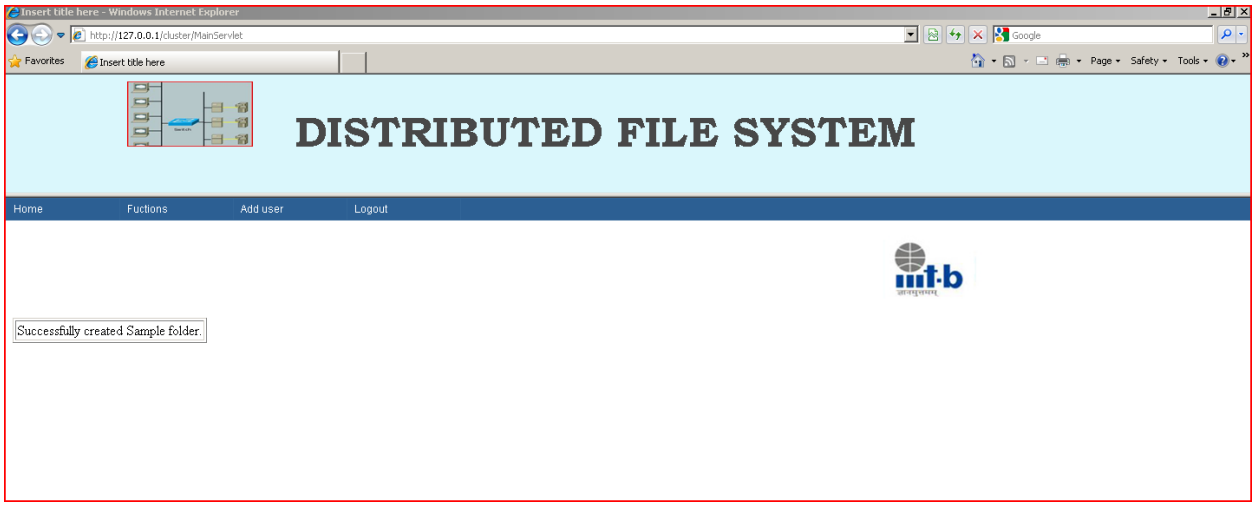
5. Click on Create Directory

The following screen appears:



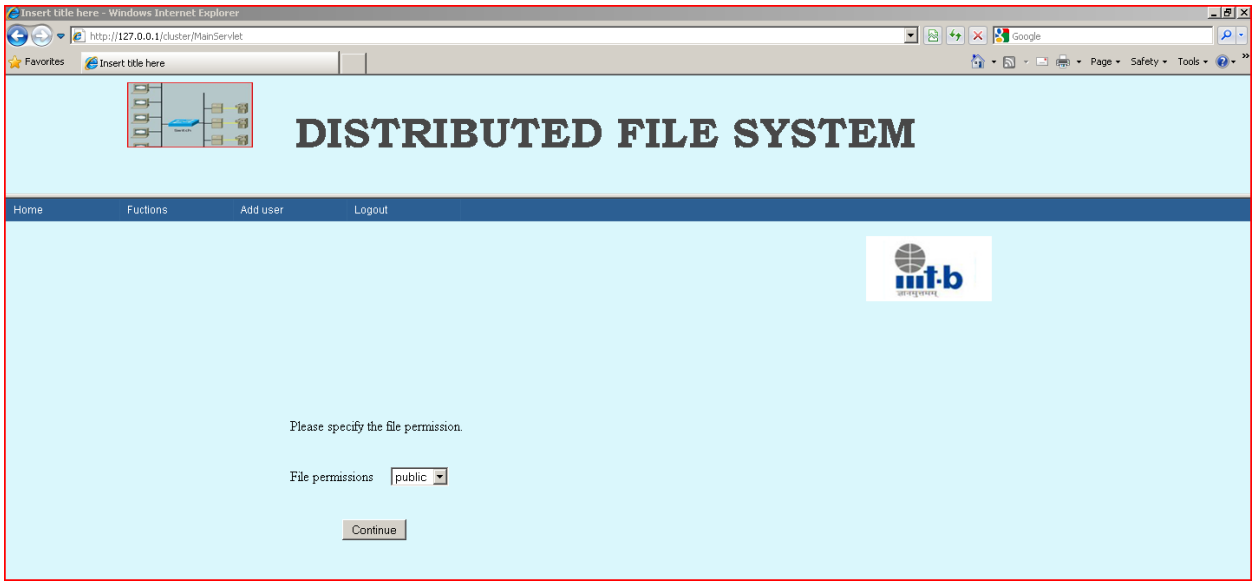
Enter the Name of the directory (for ex., Sample) in **Directory Name:** text field. Choose the permissions among public or private. The default choice is **public**.

The following screen appears:

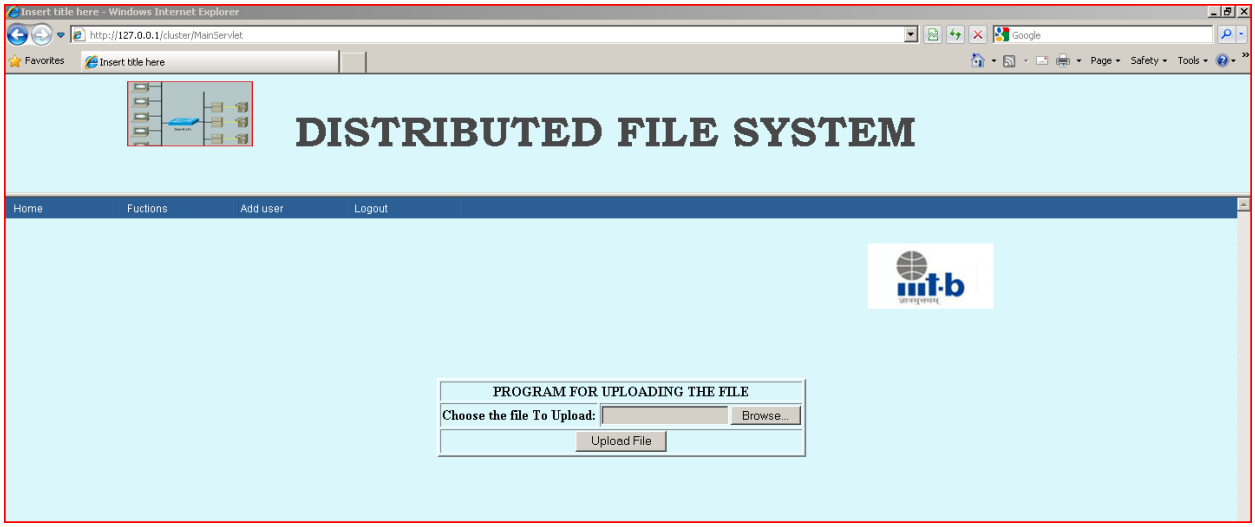


6. Click on **Upload File**

The following screen appears:



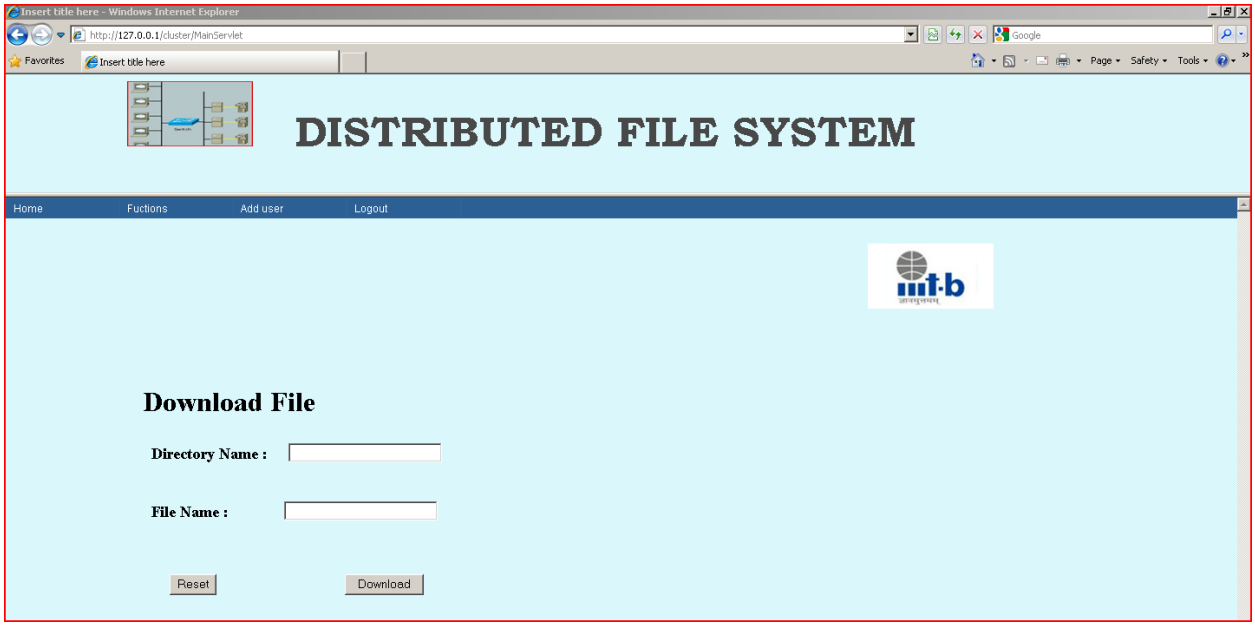
Click on **Continue** after choosing file permissions



Click on the **Browse…** button to browse for desired file. After choosing a file to upload, Click on **Upload File** button.

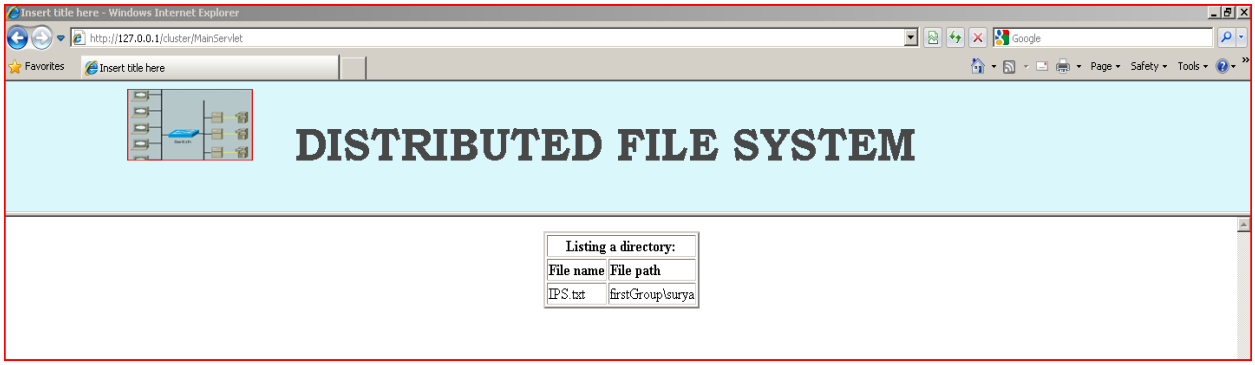
7. Click on **Download File**

The following screen appears.



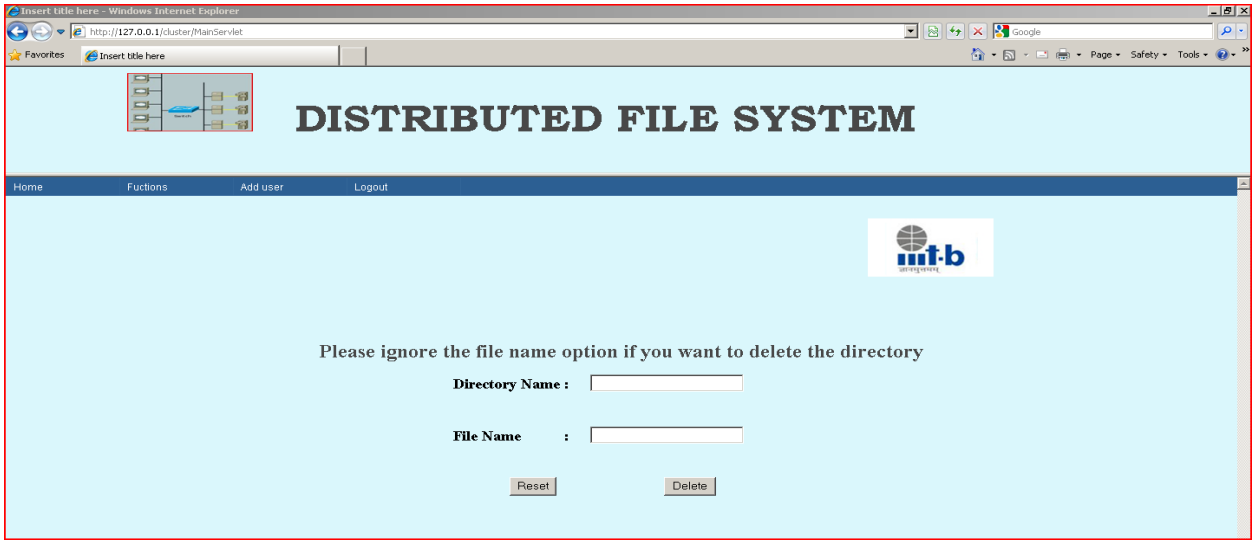
8. Click on **List Files**

The following screen appears:



9. Click on **Delete File**.

The following screen will appear:



Appendix:

Configuration of Apache and Apache-Tomcat:

This appendix explains the configuration of Apache and Tomcat servers:

Since April 1996 Apache has been the most popular HTTP server on the World Wide Web. As of March 2009 Apache served over 46% of all websites and over 66% of the million busiest. Apache is primarily used to serve both static content and dynamic Web pages on the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides. Apache is the web server component of the popular LAMP web server application stack.

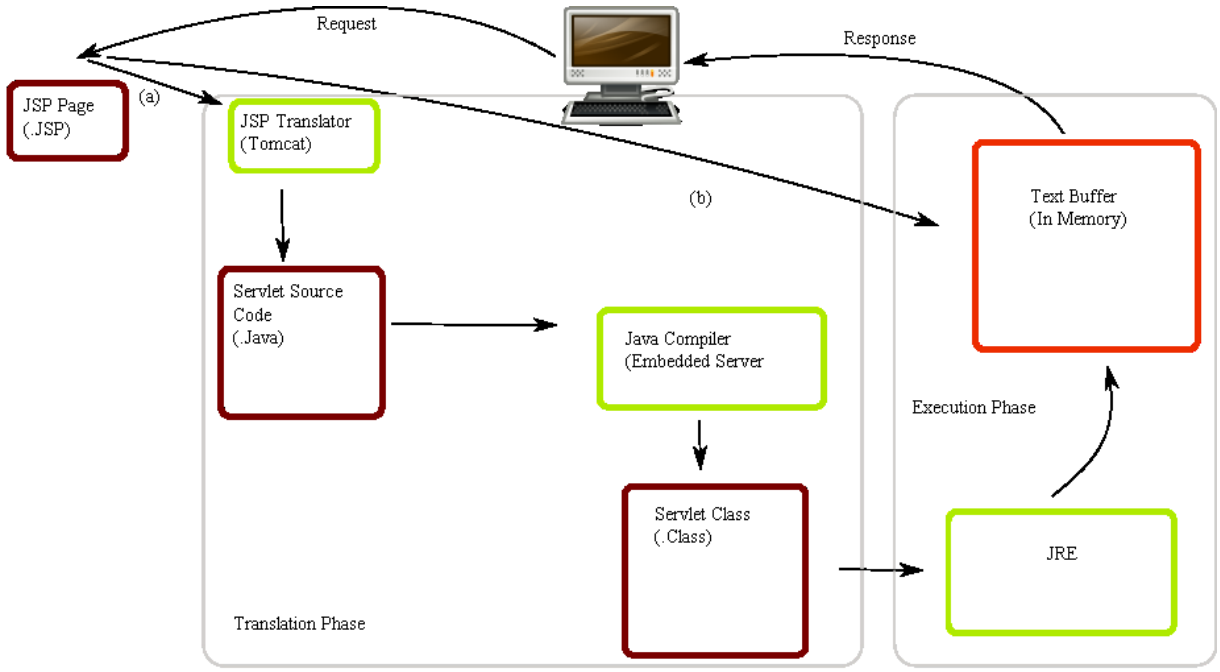
Servlets are Java programming language objects that dynamically process requests and construct responses. The Java Servlet API allows a software developer to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets are the Java counterpart to non-Java dynamic Web content technologies such as PHP, CGI and ASP.NET. Servlets can maintain state across many server transactions by using HTTP cookies, session variables or URL rewriting.

The Servlet lifecycle consists of the following steps:

- The Servlet class is loaded by the container during start-up.
- The container calls the init() method. This method initializes the servlet and must be called before the servlet can service any requests. In the entire life of a servlet, the init() method is called only once.
- After initialization, the servlet can service client requests. Each request is serviced in its own separate thread. The container calls the service() method of the servlet for every request. The service() method determines the kind of request being made and dispatches it to an appropriate method to handle the request. The developer of the servlet must provide an implementation for these methods. If a request for a method that is not implemented by the servlet is made, the method of the parent class is called, typically resulting in an error being returned to the requester.
- Finally, the container calls the destroy() method that takes the servlet out of service. The destroy() method like init() is called only once in the lifecycle of a Servlet.

A Servlet container is a specialized web server that supports Servlet execution. It combines the basic functionality of a web server with certain Java/Servlet specific optimizations and extensions – such as an integrated Java runtime environment, and the ability to automatically translate specific URLs into Servlet requests. Individual Servlets are registered with a Servlet container, providing the container with information about what functionality they provide, and what URL or other resource locator they will use to identify themselves. The Servlet container is then able to initialize the Servlet as necessary and deliver requests to the Servlet as they arrive. Many containers have the ability to dynamically add and remove Servlets from the system, allowing new Servlets to quickly be deployed or removed without affecting

other Servlets running from the same container. Servlet containers are also referred to as web containers or web engines.



JSP Container
(a) Translation occurs at this point if JSP has been changed or is new
(b) if not, translation is skipped.

Apache Tomcat is a servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run.

A WAR file (which stands for "web application archive"). There are special files and directories within a WAR file. The /WEB-INF directory in the WAR file contains a file named web.xml which defines the structure of the web application. If the web application is only serving JSP files, the web.xml file is not strictly necessary. If the web application uses servlets, then the servlet container uses web.xml to ascertain to which servlet a URL request is to be routed. web.xml is also used to define context variables which can be referenced within the servlets and it is used to define environmental dependencies which the deployer is expected to set up. An example of this is a dependency on a mail session used to send email. The servlet container is responsible for providing this service. The /WEB-INF/classes directory is on the ClassLoader's classpath. This is where .class files are loaded from when the web application is executing. Any JAR files placed in the /WEB-INF/lib directory will also be placed on the ClassLoader's classpath.

Components of Apache-Tomcat:

Catalina

Catalina is Tomcat's servlet container. Catalina implements Sun Microsystems' specifications for servlet and Java Server Pages (JSP).

Coyote

Coyote is Tomcat's HTTP Connector component that supports the HTTP 1.1 protocol for the web server or application container. Coyote listens for incoming connections on a specific TCP port on the server and forwards the request to the Tomcat Engine to process the request and send back a response to the requesting client.

Jasper

Jasper is Tomcat's JSP Engine. Tomcat 5.x uses Jasper 2, which is an implementation of the Sun Microsystems's Java Server Pages 2.0 specification. Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper is able to automatically detect JSP file changes and recompile them.

Configuration of Apache Web – Server:

Apache server is configured as a proxy server acting as a moderator between the Apache-Tomcat server(s) and client(s) connecting through a browser. The worker(s) properties of the Apache server are mentioned in workers.properties file. A sample worker property file is shown below:

```
worker.list=tomcatA,tomcatB,tomcatC,loadbalancer
worker.tomcatA.port=8109
worker.tomcatA.host=192.168.15.101
worker.tomcatA.type=ajp13
worker.tomcatA.lbfactor=1

worker.tomcatB.port=8209
worker.tomcatB.host=127.0.0.1
worker.tomcatB.type=ajp13
worker.tomcatB.lbfactor=1

worker.tomcatC.port=8309
worker.tomcatC.host=127.0.0.1
worker.tomcatC.type=ajp13
worker.tomcatC.lbfactor=1

worker.loadbalancer.type=lb
```

worker.loadbalancer.balanced_workers=tomcatA,tomcatB,tomcatC

worker.loadbalancer.method=Busyness

worker.loadbalancer.sticky_session=1

Sticky_session parameter enables the session information to be stored in the moderator so that the same session can be continued on another server in the case of failure of first server.

The apache Daemon service is configured using the httpd.conf file in Apache conf directory. Several other key parameters are also mentioned in this file:

ServerRoot "C:\Program Files\Apache Software Foundation\Apache2.2"

JkWorkersFile conf/workers.properties

JkLogFile logs/mod_jk.log

JkLogLevel error

JkMount /cluster loadbalancer

JkMount /cluster/* loadbalancer

JkMount /DFS loadbalancer

JkMount /DFS/* loadbalancer

The above locations are mounted to enable monitoring by the loadbalancer using jkMount parameter specifier.

Apache-Tomcat is configured by editing its configuration settings. Tomcats are made to work in a cluster using distributable property in web.xml file and each tomcat server is pre-configured to follow specified protocol over a port :

<Connector port="8081" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

<Connector port="8109" protocol="AJP/1.3" redirectPort="8443" />

The tomcat servers are first started by executing startup.bat file in bin directory of each tomcat server.

Followed by successful start of tomcat servers, Apache web-server is started by executing http.exe Daemon service from the bin directory of Apache.

VFS uses MySQL database to store user information along with respective privileges. SQL server is prepared using the .sql file in batch mode.